CEWES MSRC/PET TR/98-30

Identifying Boundary Anomalies to Facilitate Correct Parallel Image Composition

by

Lance C. Burton
Raghu Machiraju
Donna S. Reese

DoD HPC Modernization Program
Programming Environment and Training
CEWES MSRC

CEWES MSRC

Nichols Research

05h00298

Work funded by the DoD High Performance Computing Modernization Program CEWES Major Shared Resource Center through

Programming Environment and Training (PET)

Supported by Contract Number: DAHC 94-96-C0002
Nichols Research Corporation

Submitted to 1998 Eurographics Workshop on Parallel Graphics and Visualization

Identifying Boundary Anomalies to Facilitate Correct Parallel Image Composition

Lance C. Burton
Raghu Machiraju
Donna S. Reese
NSF Engineering Research Center for Computational Field Simulation
Mississippi State University
{burtonl,raghu,dreese}@erc.msstate.edu

Abstract

Parallel image composition presents an attractive approach to run-time visualization of structured grid data. However, certain configurations of grid boundaries prevent composition from being performed correctly. In particular, when the boundary between two partitions contains concave sections, the partitions may no longer be depth sorted correctly, a requirement for some visualization techniques such as raycasting. If the data may be repartitioned such that it can be depth sorted correctly, then an image composition approach is a viable option. To facilitate such an operation, we present an algorithm to analyze the geometric structure of a grid boundary and extract knowledge about how the boundary impacts depth sorting and therefore image composition.

1 Introduction

Parallel techniques provide researchers in many disciplines with the capability to handle large problems in a timely fashion. In computational field simulation (CFS) research, the entire pipeline may be parallelized, beginning with grid generation, continuing through computational fluid dynamics (CFD) flow solutions, and terminating with visualization. Typically, visualization is performed as a post-processing step on the output of the flow solver. Some systems handle run-time streams, but they are still logically post-mortem.

With the ever-increasing power of parallel technology, scientific applications generate datasets too large to be interactively viewed with a uniprocessor visualization system, particularly when time-consuming techniques such as direct volume rendering are employed. As with other memory- and cpu-intensive operations, parallelization provides a solution. Many parallel visualization efforts have yielded satisfactory results, again in a post-mortem fashion (Ellsworth 1993) (Ma et al. 1993) (Lacroutte 1995) (Ma 1995) .

If our application delivers results quickly enough, it may be desirable to visualize said results immediately. We might be interactively steering a simulation, debugging an algorithm, or performing a time-critical operation where (near) instant feedback is required. Given a parallel solver and a parallel visualization tool, a logical course of action to investigate would be to combine the two, performing visualization in-place alongside the solver.

Volume visualization techniques such as raycasting generate image data via a front-to-back traversal of the data (Ma et al. 1993). As a ray travels through the volume, data samples are accumulated with a constant frequency. When the technique runs in parallel, rays that traverse multiple partitions become segmented and must be gathered together for assembly. When such an action is performed by accumulating all segments on a single node, the intermediate storage requirements for the segments and the adjacency information can become quite large and hardware support for such operations is quite scarce.

Image composition presents an attractive alternative to segmented raycasting as an in-place parallel rendering scheme. In an image composition scheme, each node generates a complete image based upon its local data. The images are then globally combined, pixel by pixel, to create the final image (Porter and Duff 1984) (Duff 1985) (Molnar et al. 1994) (Lee, Raghavendra and Nicholas 1995).

The input data remains on its computational node, therefore communication complexity is divorced from data complexity. Instead, the required bandwidth depends only upon the desired resolution of the rendered images, and even that value may be lessened through optimization. Additionally, ongoing research strives to create a hardware solution for image composition (Molnar 1991) (Molnar, Eyles and Poulton 1992).

Image composition is not without its drawbacks, however. When the underlying rendering technique requires a front-to-back sorting, such as alpha blending for raycasting or transparency, an image composition approach will fail to give correct results if the data cannot be properly sorted. Such a case occurs in volumetric datasets when the data decomposition creates non-planar boundaries, such as that found in curvilinear structured grids. While in some cases a standard planar subdivision such as block or octree is appropriate for a given application, a more domain-specific decomposition, tailored to expected results, may be optimal for some parallel solvers. Under such circumstances, correct image composition is impossible because the concave nature of a non-planar boundary induces a cycle in the visibility graph, thereby precluding an unambiguous depth sort.

In *Figure 1* we see one block of a structured grid that has been partitioned radially and circumferentially. The boundaries have been adapted to accommodate the expected flow across a set of turbomachinery fan blades. The result is concavity in both the radial and circumferential boundaries. Since another block fits snugly against each of these boundaries, the aforementioned depth sorting problem occurs.

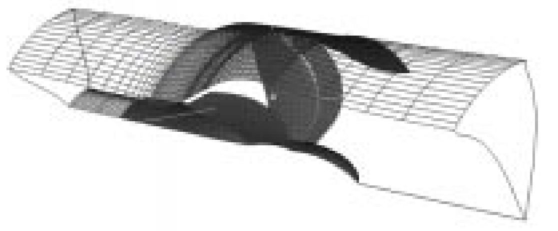

Figure 1 **One partition of a structured grid with concave boundaries**

All is not lost, however. If some partitioning exists that allows proper depth sorting for a given viewpoint, then we can always repartition and redistribute the data to perform the visualization step. Unfortunately, this approach may incur a heavy penalty, particularly when a significant percentage of the data migrates. Furthermore, the same prohibitive memory constraints that motivated a parallel solution for large datasets may not allow additional data to be stored on a given node.

Suppose instead that an acceptable partitioning exists as a superset of the current decomposition. The original demarcation remains, and further subdivision is performed on each contiguous partition. We would intuitively expect said subdivision to occur in the neighborhood of the aforementioned non-planar boundaries. The result is a set of contiguous volumes, some coexisting on a single processing node, that can be unambiguously depth sorted. Consequently, we can generate an image for each of the new partitions and composite those images accordingly, preferably with a hardware composition network that permits the computational nodes to continue with the simulation (Molnar 1991) (Molnar et al. 1992).

Pursuant to such a view-dependent dynamic partitioning, we need to extract knowledge from the topology of the data regarding boundary anomalies. While simple detection of problem areas is sufficient to indicate that a potential conflict exists and where it will occur, higher level structural information is required in order to determine the interaction between a given viewpoint and an anomaly.

The knowledge extraction algorithm consists of a sequence of identification and classification steps. We must first identify which areas of the boundary will pose an obstacle to depth sorting. This is done by computing the difference between the grid surface and the convex hull of the grid. The difference will be comprised of zero or more discrete contiguous volumes, or cavities. Having identified the cavities, we then perform an edge detection step to further subdivide each cavity into entirely convex or entirely concave patches. We do this as part of a divide-and-conquer scheme. The patches are then analyzed independently to determine the effect of each upon composition when considered separately. Finally, we calculate the interaction among groups of groups of patches and combine that with the individual information to specify the overall behavior in terms of what sections of the grid will present problems for viewpoints in characteristic zones of view space.

In Section 2 and Section 3 we investigate the initial stages of the knowledge extraction process, including cavity detection and classification. Section 4 explores the local impact of a single monotonically convex or concave patch upon image composition, while Section 5 expounds upon the relationship between groups of patches. Section 6 pulls the results from the previous sections into a single coherent classification of view space into characteristic zones. Concluding remarks and the direction of future work are presented in Section 7.

2 Cavity Identification

The first step in identifying boundary anomalies is to find all discrepancies between the convex hull and the grid area/volume. These anomalies permit lines of sight to enter and exit a volume multiple times. Such discrepancies manifest themselves in three forms (Williams 1992):

cavity - a “dimple” in the boundary, formed by the enclosure of the convex hull and a contiguous subsection of the grid boundary that is not part of the convex hull

void - an internal vacuum, formed solely by a closed, contiguous section of the surface not part of the convex hull

hole - a puncture through the volume, formed by a closed, contiguous section of the surface not part of the convex hull and two disjoint but locally contiguous subsections of the convex hull (only meaningful in 3D)

For purposes of this paper, we will confine our attention to the most common deformation, the cavity. Since, by definition, those sections of the surface that belong to the convex hull are unable to “see” any other sections, we can safely ignore them for purposes of depth sorting. However, we may use the convex hull itself to assist in distinguishing the regions on the hull from the cavities.

Consider the partial structured grid in *Figure 2*. Three boundaries are depicted, two of which are convex. The third is a Beziér curve such as one might expect as the result of a computer-aided design effort. Depending upon the application, a parallel solver may find it advantageous to use such a curve as a basis for data decomposition. Note that the curve is concave in some regions, preventing unambiguous depth sorting should another partition lie along that boundary

Well-known algorithms exist to find 2D and 3D convex hulls for an arbitrary set of points (Cormen, Leiserson, and Rivest 1990) (Edelsbrunner and Shi 1991). In *Figure 3*, we see the convex hull of the sample grid from *Figure 2*. Note that the hull follows along the two convex edges on either side and stretches across the concave edge, forming a cavity. In 3D, the hull would be some sort of non-planar surface, composed of twisted quadrilaterals.

The cavity itself is defined by two surfaces, as mentioned previously. For the cavity in *Figure 3*, each surface is defined by a set of discrete points, as shown in *Figure 4*.

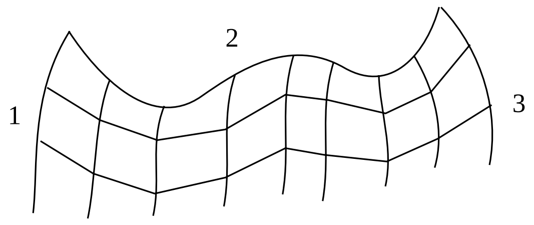

Figure 2 **A concave boundary of a 2D structured grid - boundary 2 is concave**

Identification of these defining points combined with knowledge of the implicit connectivity of the grid allow us to reconstruct the cavity, which will be used in later algorithms.

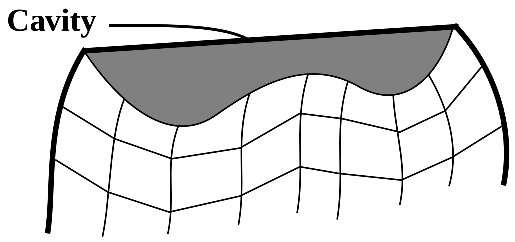

Figure 3 **The local convex hull for *Figure 2***

For a 2D structured grid, one straightforward approach is to walk the convex hull, in either a clockwise or counterclockwise fashion, and mark where the grid surface deviates from the hull. Another method is to group all surface points not on the convex hull, select one, and grow a concave region around it. For this research, we choose the latter because the algorithm translates directly to use on a 3D structured grid.

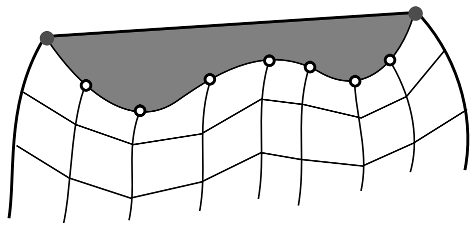

Figure 4 **Defining points of the cavity**

The function cavityWalk takes as input a grid specification G and returns a cavity list CL. Once the convex hull has been calculated, an exterior point not on the hull is selected, and the cavity is recursively grown around that point. The cavity is grown by recursing on all neighboring points, with an anchor at either a point already in the cavity or a member of the convex hull. The notion of a neighbor is very important. In order to properly include corner points, we need to look at 8-way neighbors on the surface, not just 4-way. In some cases, an extreme edge - an edge of the grid's cube in parametric space - may actually not be part of the convex hull, therefore we must consider neighbors on adjacent "walls" as well. The region growing stops when a boundary consisting entirely of points on the convex hull has been reached. Once a cavity has been identified and stored in the list, a new point is selected that again is not on the convex hull and is also not part of any cavity discovered so far. The process continues until all exterior points have been assigned a cavity.

```
function cavityWalk( grid G, cavityList CL )
{
    compute convex hull CH(G)
    let E be all exterior points of G not in CH(G)
    while ( |E| > 0 )
    {
        select a point p from E
        remove p from E
        assign p to a new cavity C
        grow( p, C )
        add C to CL
    }
}

function grow( point p, cavity C )
{
    for ( each surface neighbor n of p )
    {
        if ( !( n ∈ C) )
        {
            add n to C
            if ( !(n ∈ CH(G)) )
            {
                remove n from E
                grow( n, C )
            }
        }
    }
}
    }
}
```

3 Convex/Concave Classification

To further assist the knowledge extraction process, we apply a divide-and-conquer methodology to break the problem into smaller, more manageable pieces. Given the ongoing conflict between convex and concave regions, we dissect and classify the entire cavity according to this dichotomy. The overall cavity is filtered through a kind of edge detection algorithm that separates the surface into monotonically convex or concave patches, defined as follows:

Definition: *Monotonically convex/concave* - a curve C is monotonically concave or convex if for all points $p \in C$, curvature(C,p) has the same orientation or is zero

In the continuous domain, a curve may only change convexity (with respect to a certain notion of inside and outside) where its curvature becomes zero. Obviously, a convexity change requires that the curve be locally convex on one side of the point of zero curvature and locally concave on the other side. Furthermore, extended regions of zero curvature may exist.

For a discrete curve, such as our structured grid boundary, regions of exact zero curvature may not exist, due to the inherent nature of the sampling process. However, the curve is piece-wise linear along grid lines. Consequently, the local curvature may be easily calculated by examining the crease angle between adjacent facets. Angles of less than 180 degrees indicate concavity. Therefore, a monotonically convex region of a discrete curve is one whose component crease angles are all 180 degrees or more, while a monotonically concave region contains only crease angles of less than 180 degrees.

We may therefore use the crease angles within a region to classify said region as convex or concave, provide all crease angles within the region are designated with the same orientation. The boundaries of each region are defined by transition points (edges in 3D). A transition point itself is either a concave or a convex junction, and is neighbored on one side by a concave junction and on the other by a convex junction. Furthermore, each point of the cavity belonging to the convex hull bounds one or more regions. The transition points for the cavity in *Figure 4* are shown in *Figure 5*.

The actual placement of transition points is subject to some interpretation for a discretely sampled curve. Transitions on such a curve actually happen across a face, the

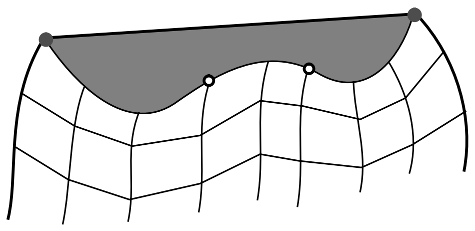

Figure 5 **Transition points between monotonically convex and concave regions**

boundaries of which become candidates for actual transition points. *Figure 6* shows an adjacent convex-concave pair of patches that could be separated at one of two junctions, one concave and one convex.

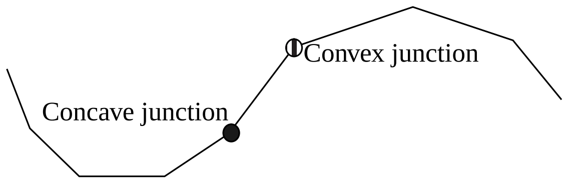

Figure 6 **Possible transition points**

Depending upon the actual layout of the surrounding points, a given selection of a transition point may actually remove a patch of sufficiently small size, e.g., a patch formed of only three points. Such small patches are effectively absorbed into neighboring patches. The exact benefit or detriment of such absorption is unclear, but it seems intuitively to be positive. Furthermore, regions of zero curvature within a cavity present an additional factor to consider. For example, consider the stairstep pattern in *Figure 7*. Selecting the hollow circle in the center as a transition point yields two concave regions, but selecting two points, one from each of the arms connected to the hollow circle, yields an additional convex region. In order to select the best set of transition points, we need a set of guidelines or heuristics to assist the process.

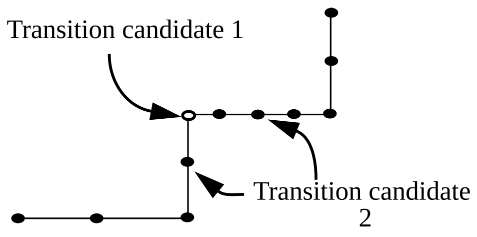

Figure 7 **Transition point ambiguity**

Once we have actually selected the transition points, we can then group points into connected components, where each connected component consists of a set of transition points T plus all intermediate points P such that for any point $p \in P$, there exists a path from p to each point in T that does not pass through any transition point not in T, and there exists no path from p to a transition point not in T that does not pass through a transition point that is in T. Such conditions ensure that each non-transition point in the cavity belongs to exactly one connected component and each transition point belongs to at least one. In practice, an incremental region growing algorithm that works its way from transition point to transition point should satisfy the requirements. A classification for the cavity in *Figure 4* is shown in *Figure 8*.

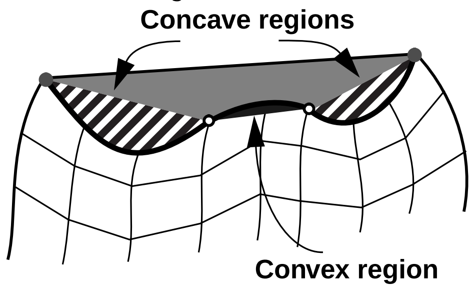

Figure 8 **Classified monotonically convex and concave patches**

4 Intraregion Influence

Once we have classified each convex and concave region within the cavity, we need to determine each region's effect on depth sorting. The correctness of any given depth sorting is dependent upon the view, but we may determine a priori a set of ranges specifying which patch(es) preclude depth sorting for a view within a given range. Regions interact not only with each other, but with themselves as well. Therefore, we will investigate intraregion influence and interregion influence separately.

The intraregion influence of a convex or flat section is obvious - by definition, a line of sight exiting such a surface will not reenter at another point on that surface. We can therefore safely confine our focus to concave regions. The configuration of a concave region directly influences the effect said region has on depth sorting. We classify a configuration by noting relative positions and orientations of the endpoints of the region's defining curve.

Consider a typical case with a small degree of curvature. In *Figure 9*, we see a simple depression that forms a kind of obtuse mouth. The tangents of the curve at the endpoints flare out, diverging in front of the mouth. If our viewpoint lies within the area bounded by the curve and the two tangent lines, then the curve will not pose an obstacle to correct depth sorting. The diametrically opposed region behind the convergence point provides similar benefits.

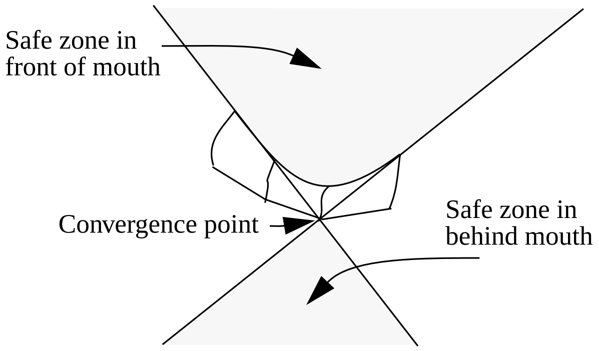

Figure 9 **Concave patch with obtuse mouth**

As the tangent lines become parallel, the convergence point, and its associated safe zone, move out to an infinite distance from the curve. *Figure 10* shows a concave region whose mouth is square, i.e. the tangent lines are parallel. Note that the only safe area lies in the rectangle-like region directly in front of the curve.

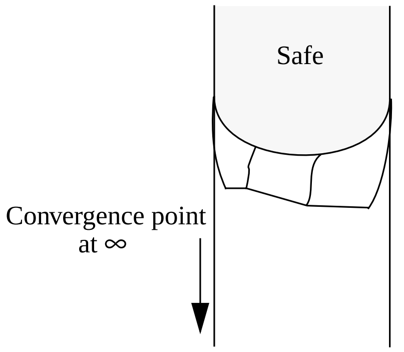

Figure 10 **Concave patch with square mouth**

Inward deviation from the parallel configuration creates an acute mouth with a convergence point that now lies in front of the curve. As a result, the safe viewpoint region, as with the square mouth in *Figure 10*, only lies in front of the mouth and is bounded by the curve itself and the two tangent lines. Such a curve is seen in *Figure 11*.

When the tips of the curve start to turn in upon the curve itself, the tangent line exiting one or more of the endpoints intersects the curve at a separate point. Once this happens, not even the area within the mouth of the curve remains safe. As seen in *Figure 12*, only the region

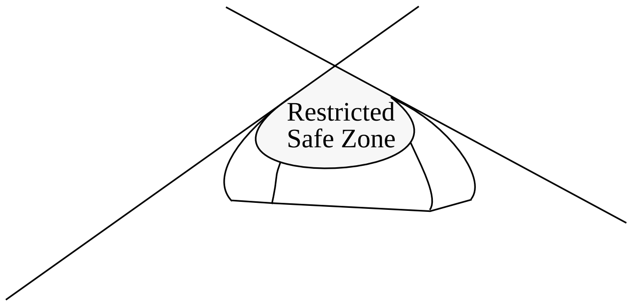

Figure 11 **Concave patch with acute mouth**

bounded by the curve and the bridge between endpoints allows a safe viewpoint.

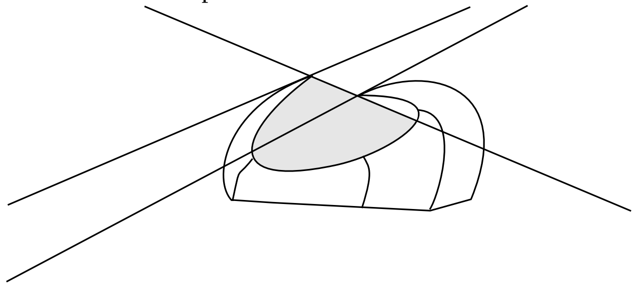

Figure 12 **Concave patch with self-intersecting tangent lines**

5 Interregion Interaction

Whether or not a curve patch poses a problem with respect to itself, its interaction with other patches must be considered as well. Any pair of patches that can “see” one another create a sorting anomaly. Two patches “see” each other if a line of sight can be drawn from the front face of one patch to the front face of the other.

A concave patch can see a convex patch, and vice versa, if and only if we can draw a line that extends from an endpoint of the concave patch and is tangent to the convex patch, and the convex patch exists in the half-space formed by the concave patch. The line condition ensures that the convex patch is facing the concave patch, and the half-space condition ensures the reverse. *Figure 13* shows examples of convex-concave pairs that fail one or both conditions and therefore cannot see each other.

If two patches can see each other, then we can determine their mutual effect by treating each defining point of a patch as a point light source and calculating the shadow cast by the other patch. This is similar to the visibility computations done by Teller that computed the antipenumbra of an area light source through a series of portals

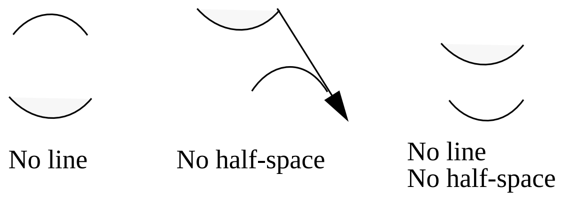

Figure 13 **Convex-concave pairs that cannot see each other**

(Teller 1992). The illumination volume from Teller's projection specified what regions of a densely occluded polyhedral environment were visible through a given portal, i.e., a gap between solid surfaces.

Similarly, we would like to ascertain two important pieces of information: (1) exactly which portions of each patch are mutually visible; and (2) the range in which a viewpoint may have a line of sight that exits one patch and enters another. The point source projection gives us both, with the proper interpretation. The resultant shadow volume bounds the range of unsafe viewpoints, while the intersection of the volume with the patch casting the shadow defines the region of visibility. The union of all such volumes and intersections fully specifies the mutual influence of two patches as well as the bounded viewpoint areas for which correct depth sorting cannot be accomplished.

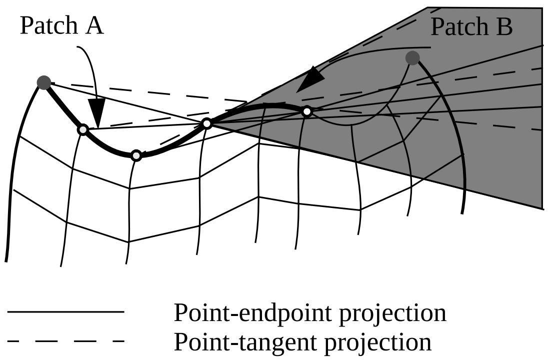

Figure 14 **Shadow volume from projecting patch A to patch B**

The shadow volume generated by projecting one point source over a patch is bounded by the set of lines tangent to the patch that intersect the point source. For our discrete structured grid, these lines will actually connect the point source to one of the defining points of the patch. *Figure 14* shows the shadow volume created by projecting each defining point of patch A over patch B. Each point source creates a conic area that overlaps some of the other projections. The aggregate overlap of all the areas defines the regions in space in which a viewpoint may lie that has at least one line of sight that will exit a front face of patch B and enter a front face of patch A.

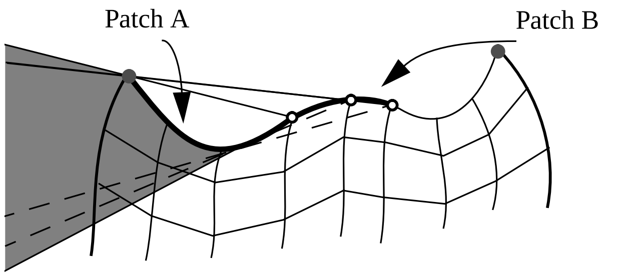

Figure 15 **Shadow volume from projecting convex patch B onto concave patch A**

The inverse projection, from patch B to A, is in *Figure 15*. Again, we have conic areas that overlap and merge to form a single contiguous area that represents our troublemakers in view space. Combining this information with that gleamed from the projection in *Figure 14* yields a dissection of space into safe regions and not safe regions, as marked in *Figure 16*

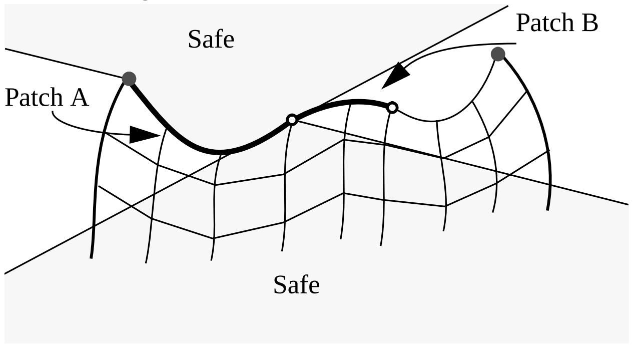

Figure 16 **Combined visibility and safe viewpoint ranges for composition with respect to adjacent patches A and B**

Performing the projection operation on the next two patches, B and C, gives us a similar situation. As we see in *Figure 17*, even the geometry of the safe and unsafe regions is comparable, due to the similarity between regions A and B. Note that both of these pairings, A/B and B/C, are of adjacent patches, one convex and one concave.

The procedure is identical for patch pairs that are convex-convex, concave-concave, and/or not adjacent. For patches A and C, we perform the same point source projection on each defining point of each patch. The resultant aggregate shadow volume again allows us to determine visibility and safe viewpoint information, as seen in *Figure 18*.

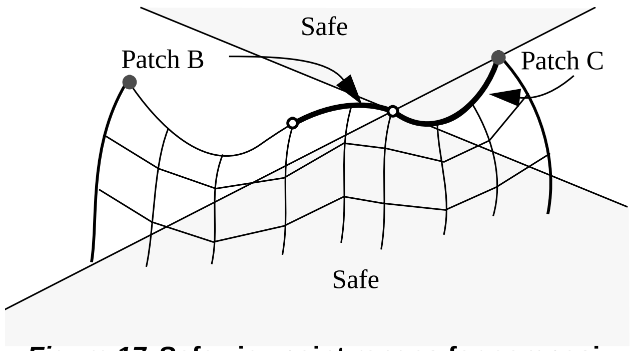

Figure 17 **Safe viewpoint ranges for composition with respect to adjacent patches B and C**

Depending on the exact configuration of the patches, the shadow volume method described above may actually delineate larger regions than necessary. Our visibility requirement stipulates that external faces must be able to see each other. In many cases, however, portions of one patch may occluded from another, either by the patch itself or by other intervening patches.

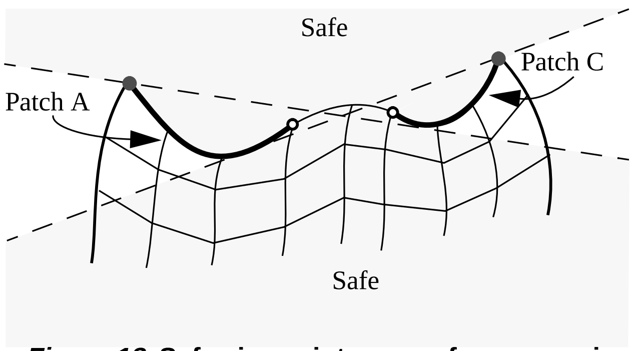

Figure 18 **Safe viewpoint ranges for composition with respect to non-adjacent concave patches B and C**

Occlusion from intervening patches will be handled by the interplay between each such patch and the current patches under consideration. Self-occlusion may be modelled by again computing shadow volumes, but only those that strike a front face of a patch. Note that previously we only considered the shadow volume on the back face of each patch by essentially projecting around the patch's convex hull.

In *Figure 19* are two such self-occluding regions, one from each of patch A and patch C. No line of sight may be drawn from A that strikes a front face of the self-occluding region in C before striking a back face, and vice versa. Therefore, we may amend our visibility information accordingly, as in *Figure 20*.

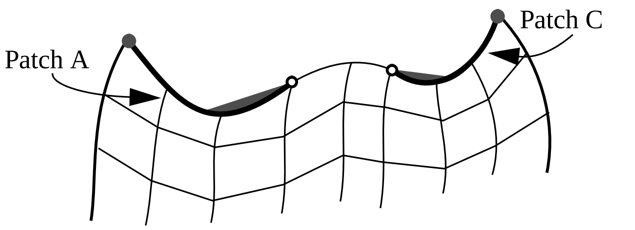

Figure 19 **Regions of self-occlusion**

6 Characteristic Zones

As we have seen in the previous sections, each monotonically convex or concave patch has its own distinct impact upon composition. The concave patches are self-occluding and therefore have their own individual signature in view space, separating the safe viewpoints from the problematic ones. However, we are trying to gauge the global behavior of the entire cavity, so we need to consider how to merge the individual solutions of our divide-and-conquer approach.

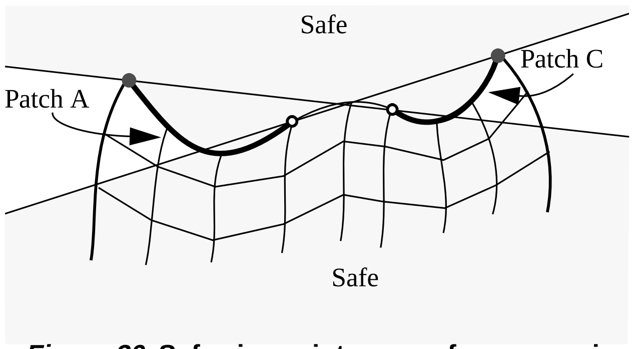

Figure 20 **Safe viewpoint ranges for composition with respect to non-adjacent concave patches B and C, considering occlusion**

In Section 4, we examined how a single concave patch affected composition and determined that a patch with an obtuse mouth sectioned view space into four quadrants, two of which were safe for composition and two of which were not. Suppose we combined the results of two of these single patch analyses. *Figure 21* shows the breakdown of space with the merged results for the concave patches from *Figure 8*.

Each of patches A and C has its four quadrants, but the quadrants overlap, further dividing the view space. The intersections of the safe zones are of course still safe, but

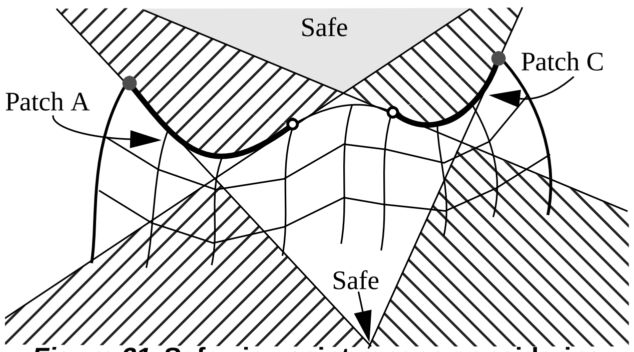

Figure 21 **Safe viewpoint ranges considering multiple intrapatch influence**

they are now much smaller. The remainder of the safe areas are divided into two categories - (1) safe with respect to A but not C; and (2) safe with respect to A but not C. The completely unsafe zones have also shrunk - pieces of them have been absorbed into the aforementioned conditional safe zones. Hence, we now have eight distinct areas of view space, each with their own characteristics. Those characteristics specify which patches will hamper composition for any viewpoint in a given area, thus allowing us to confine our repartitioning efforts to just the necessary elements.

We can extend this combination concept to include all of the extracted knowledge, including both intraregion and interregion data. The result, as above, is a dissection of view space into a collection of disjoint areas, each of which has associated visibility information about the interaction among patches. For our sample grid, we consolidate our previous five analyses (one for each of the convex patches, and one for each pairing) to produce the overall fragmentation in *Figure 22*.

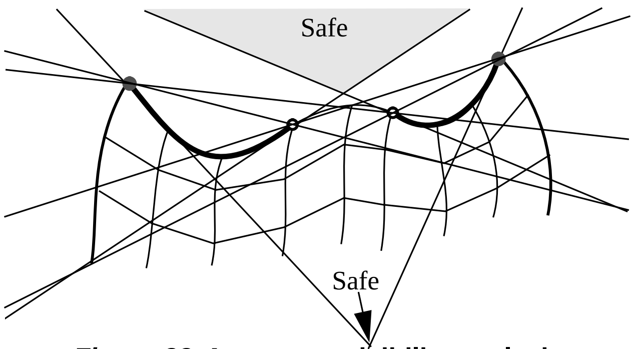

Figure 22 **Aggregate visibility analysis**

Each fragment of view space is now keyed not only to the intraregion influence but also to the interaction between patches. Therefore at any given point in space, we can determine our region and thence identify any problem patches and determine how those patches affect composition.

7 Conclusions and Future Work

We have a shown an algorithm for identifying boundary anomalies on structured grids that will inhibit parallel image composition. We have further demonstrated a methodology by which an anomaly may be broken down into its component convex and concave patches. The resultant patches of monotonic curvature may then be analyzed using shadow volumes to determine visibility information regarding which front faces of a patch present a depth sorting problem. The combination of all such analyses yields a global visibility solution that dissects the view space into disjoint regions, each of which is affected, in terms of composition correctness, by some subset of the surface patches.

Several issues need to be addressed in order to bring the eventual goal of correct image composition to fruition. First, a solid algorithm to select convex-to-concave transitions is required. A certain amount of flexibility exists in this selection due to the discrete nature of structured grids. As stated in Section 3, transitions can occur on either a concave or a convex edge. The only requirement is that the neighboring transitions parallel to the edge under consideration be one each of convex and concave. The selection flexibility comes into play particularly for very small local deformations, such as a convex peak composed of only four edges. One selection may isolate this peak while another will distribute it into the neighboring concave patches. *Figure 23* shows just such an example. To properly solve this issue, we will need to apply some heuristic techniques to identify patterns and most likely collect some empirical data to determine the quality of any given selection.

As with many geometric problems, extension from 2D to 3D incurs an immense increase in complexity. Whereas in 2D we can define our structures in terms of simple points and curves, a 3D representation requires curves and surfaces, respectively. In our small 3D example of *Figure 23*, the patch surfaces are separated by a discrete closed curve. In this example, the transition curve happens to be of a nice roughly rectangular shape. A real grid, such as the turbomachinery section in *Figure 24*, may deliver transition curves with jagged edges, similar to the aliasing found in line drawing algorithms. The curves themselves may even be concave.

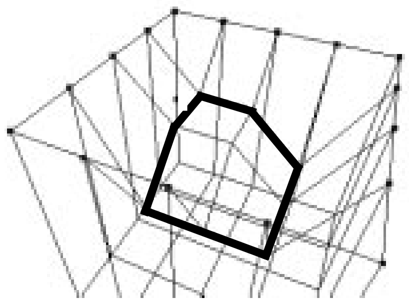

Figure 23 **3D cavity with multiple transition options**

Aside from the representation, the basic algorithms remain the same for 2D and 3D. For example, the cavity identification algorithm outline in Section 2 translates directly to 3D. The primary differences lie in the tools used to build the algorithm - the convex hull algorithm, the collection of "surface" points, and the notion of neighbors. When calculating transitions, 3D patches are bounded by curves. One notable issue here is that some patches may have holes, such as when a depression has a small convex bump in the middle. The shadow volumes generated in Sections 4 and 5 become true volumes bounded by complex surfaces, rather than the simple conic sections shown in the 2D examples, and their unions and intersections become correspondingly more complex.

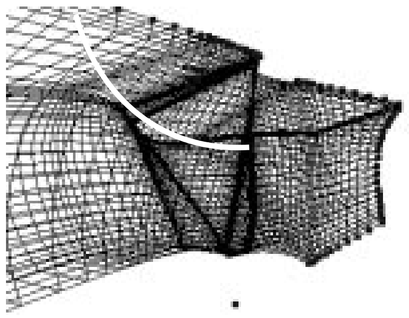

Figure 24 **Turbomachinery grid with identified cavity and expected transition curve**

The current implementation inputs PLOT3D files (Bancroft et al. 1991) and calculates the convex hull, identifies cavities, and classifies each edge as concave or convex. As stated above, a smart algorithm is required to properly delineate the convex patches from the concave ones. In *Figure 24*, *Figure 25*, and *Figure 26* we see different views of the partition of the turbomachinery grid from *Figure 1*. The points along the cavity boundaries are identified by the square markers. By definition, each of these points belongs to the convex hull. Also shown are the expected transition curves for this grid. Despite the length-wise wave effect of the grid, only one true convex patch exists on the surface. The rest of the surface that is convex length-wise is concave in the orthogonal direction, forming a saddle.

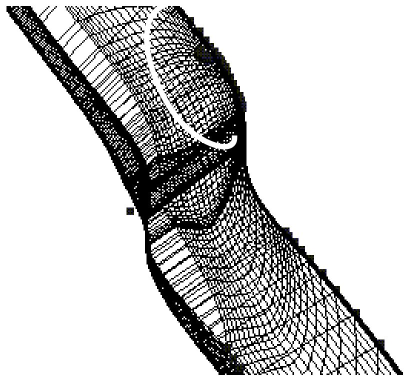

Figure 25

Once we have handled all of the issues regarding transition points and 3D representation, our next step is to determine how to use the information extracted thus far to repartition our data in such a fashion as to allow correct depth sorting and consequently correct image composition. The regions portrayed in *Figure 22* provide a good starting point. For any given viewpoint, we can determine which regions are problematic and the extent to which they are so. This same information will prove useful in determining how to repartition our data at run-time. Given that the results of our algorithm delineate view space into characteristic zones, one approach we may take is to determine a characteristic view for each zone that embodies the information necessary to repartition (Weinshall and Werman 1997).

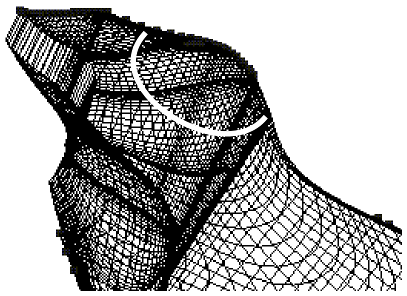

Figure 26

Acknowledgements

This work was supported by the NSF Engineering Research Center for Computational Field Simulation, Mississippi State University and the DoD High Performance Computing Modernization Program CEWES Major Shared Resource Center through Programming Environment and Training (PET) (Supported by Contract Number: DAHC 94-96-C0002 Nichols Research Corporation).

References

[1] Bancroft G., F. Merritt , T. Plessel, P. Kelaita, R. McCabe, and A. Globus. 1991. FAST: A Multi-Processed Environment for Visualization of Computational Fluid Dynamics. AIAA Paper 91-0793, In Proceedings of the 29th Aerospace Sciences Meeting, Reno, NV.

[2] Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. 1990. Introduction to Algorithms. Cambridge: The MIT Press.

[3] Duff, Tom. 1985. Compositing 3-D Rendered Images. In Proceedings of ACM SIGGRAPH '85 : 41-44.

[4] Edelsbrunner, H. and W. Shi. 1991. An $O(n \log 2 h)$ Time Algorithm for the Three-dimensional Convex Hull Problem. SIAM J. Comput. 259-277.

[5] Ellsworth, David. 1993. A Multicomputer Polygon Rendering Algorithm for Interactive Scientific Visualization. In Proceedings of the 1993 Parallel Rendering Symposium in San Jose, CA, 1993, 43-48.

[6] Foley, James D., Andries vanDam, Steven K. Feiner, and John F. Hughes. 1990. Computer Graphics Principles and Practice. Reading: Addison-Wesley.

[7] Lacroutte, Phillipe. 1995. Real-Time Volume Rendering on Shared Memory Multiprocessors Using the Shear-Warp Factorization. In Proceedings of the 1995 Parallel Rendering Symposium in Atlanta, GA, 23-30.

[8] Lee, Tong-Yee, C.S. Raghavendra, and John Nicholas. 1995. Image Composition Methods for Sort-Last Polygon Rendering on 2-D Mesh Architectures. In Proceedings of the 1995 Parallel Rendering Symposium in Atlanta, GA, 55-62.

[9] Ma, Kwan-Liu, James Painter, Charles Hansen, and Michael Krogh. 1993. A Data Distributed, Parallel Algorithm for Ray-traced Volume Rendering. In Proceedings of the 1993 Parallel Rendering Symposium in San Jose, CA, 15-22.

[10]Ma, Kwan-Liu. 1995. Parallel Volume Ray-Casting for Unstructured-Grid Data on Distributed-Memory Architectures. In Proceedings of the 1995 Parallel Rendering Symposium in Atlanta, GA, 23-30.

[11]Molnar, Steven. 1991. Image-Composition Architectures for Real-Time Image Generation. Ph.D. dissertation, University of North Carolina at Chapel Hill.

[12]Molnar, Steven, John Eyles, and John Poulton. 1992. PixelFlow: High-Speed Rendering using Image Composition. In Proceedings of ACM SIGGRAPH '92 : 231-240.

[13]Molnar, Steven, Michael Cox, David Ellsworth, and Henry Fuchs. 1994. A Sorting Classification of Parallel Rendering. IEEE Computer Graphics and Applications 14 (July): 23-32.

[14]Porter, Thomas and Tom Duff. 1984. Compositing Digital Images. In Proceedings of ACM SIGGRAPH '84 : 253-259.

[15]Teller, Seth. 1992. Visibility Computations in Densely Occluded Polyhedral Environments. Ph.D. dissertation, University of California at Berkeley.

[16]Weinshall, Daphna and Michael Werman. 1997. On View Likelihood and Stability. IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (February): 97-108.

[17]Williams, Peter. 1992. Visibility Ordering Meshed Polyhedra. ACM Transactions on Graphics 11 (April): 103-126.